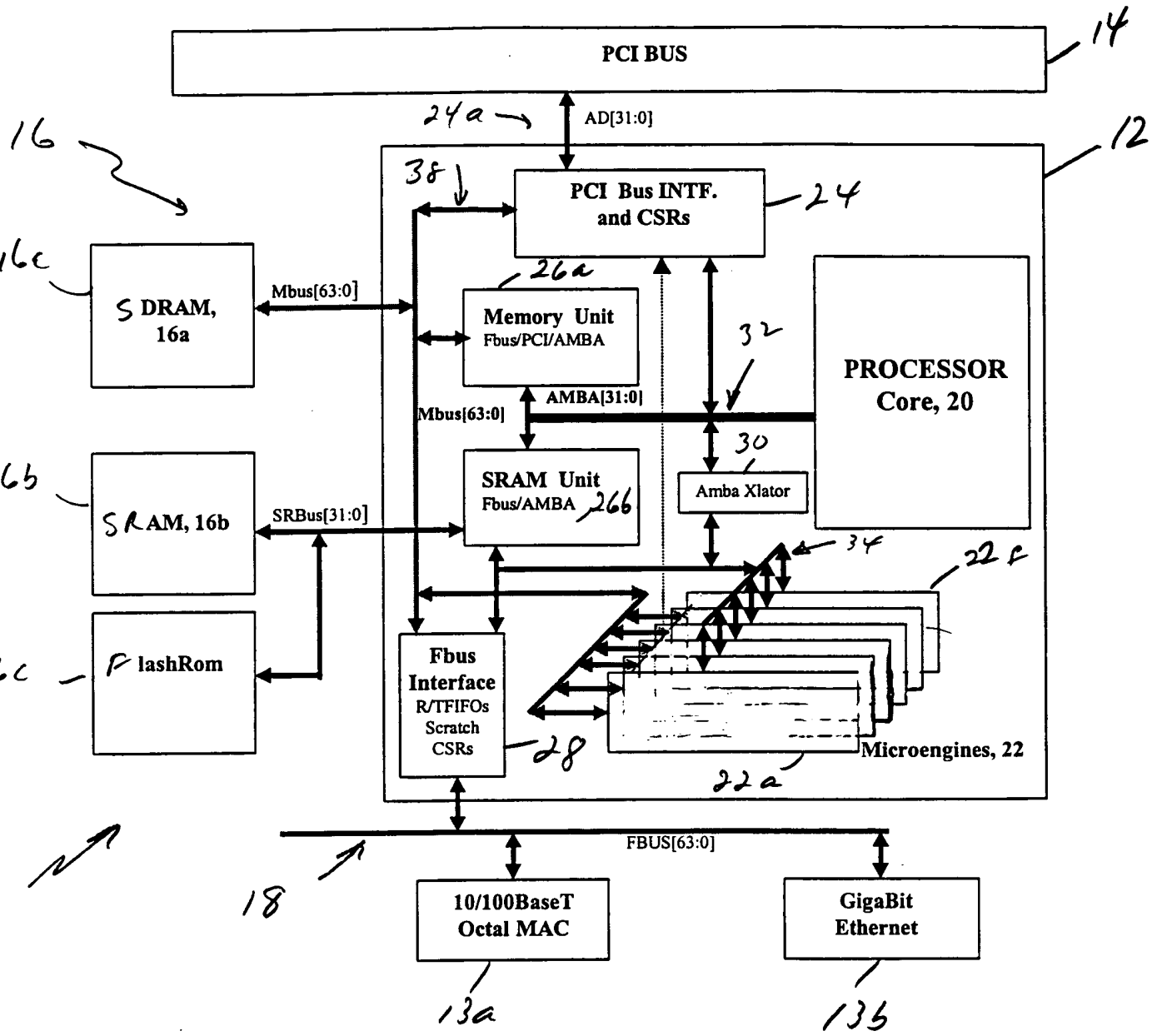

PCI BUS
14
24a
AD[31:0]
16
12
38
PCI Bus INTF.
and CSRs
24
16c
SDRAM,
16a
Mbus[63:0]
26a
Memory Unit
Fbus/PCI/AMBA
32
PROCESSOR
Core, 20
AMBA[31:0]
Mbus[63:0]
30
16b
SRAM, 16b
SRBus[31:0]
SRAM Unit
Fbus/AMBA
266
Amba Xlator
34
22c
16c
FlashRom
Fbus
Interface
R/TFIFOs
Scratch
CSRs
28
Microengines, 22
22a
FBUS[63:0]
10
18
10/100BaseT
Octal MAC
GigaBit
Ethernet
13a
13b

FIG. 1

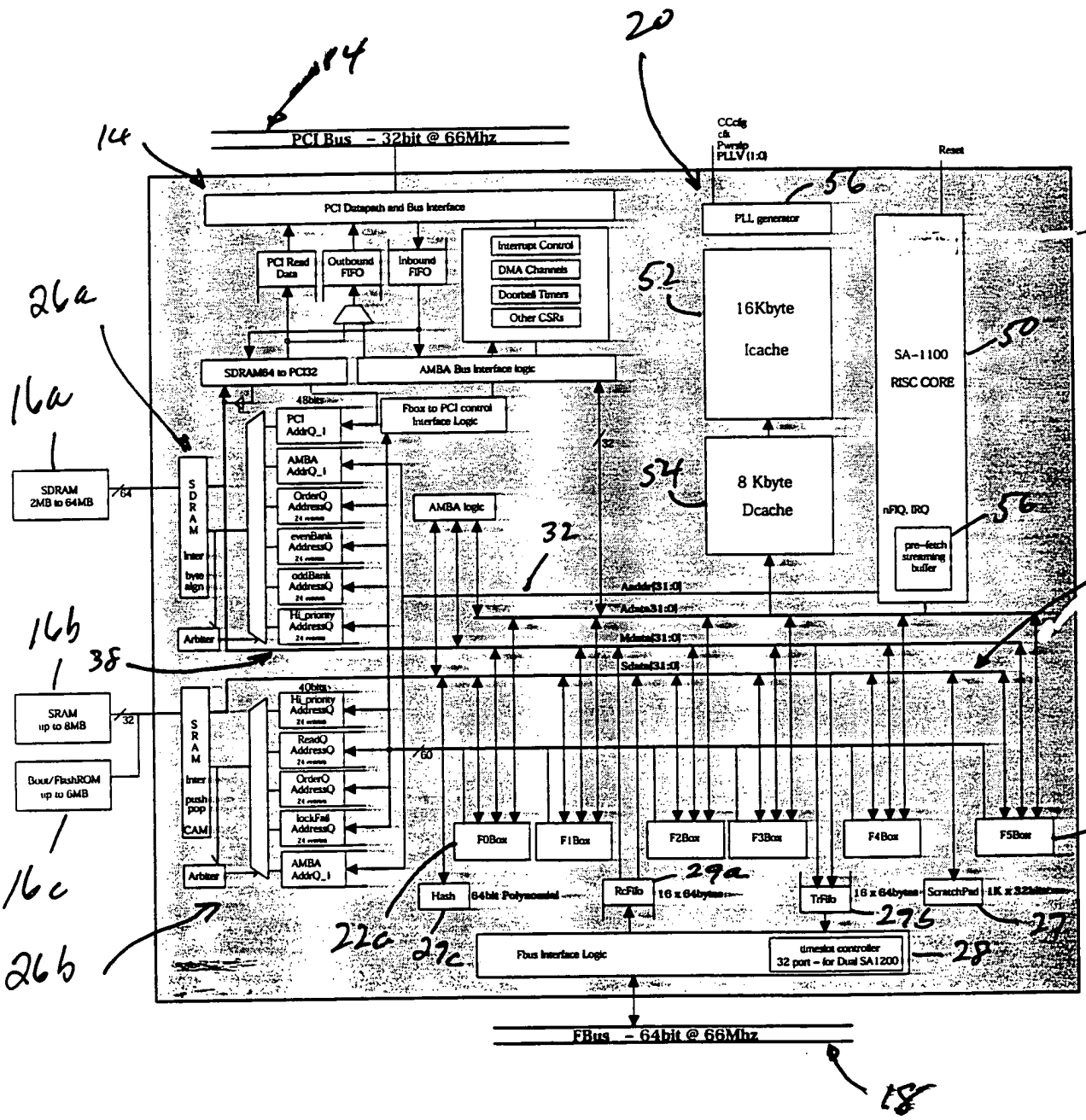

PCI Bus – 32bit @ 66Mhz
PCI Datapath and Bus Interface
PCI Read Data
Outbound FIFO
Inbound FIFO
Interrupt Control
DMA Channels
Doorbell Timers
Other CSRs
SDRAM64 to PCI32
AMBA Bus Interface logic
Fbox to PCI control Interface Logic
PCI AddrQ_1
AMBA AddrQ_1
OrderQ AddressQ
evenBank AddressQ
oddBank AddressQ
Hi_priority AddressQ
AMBA logic
Arbiter
SDRAM
SDRAM 2MB to 64MB
SRAM up to 8MB
Boot/FlashROM up to 6MB
SRAM
ReadQ AddressQ
lockFail AddressQ
CAM
PLL generator
16Kbyte Icache
8 Kbyte Dcache
SA-1100 RISC CORE
nFIQ, IRQ
pre-fetch streaming buffer
F0Box
F1Box
F2Box
F3Box
F4Box
F5Box
Hash
64bit Polynomial
RcFifo
16 x 64bytes
TrFifo
ScratchPad
Fbus Interface Logic
timeslot controller
32 port – for Dual SA1200
FBus – 64bit @ 66Mhz
14
20
16a
16b
16c
26a
26b
22a
22c
22f
29a
27b
27
28
18
32
34
38
52
54
56
50

FIG. 2

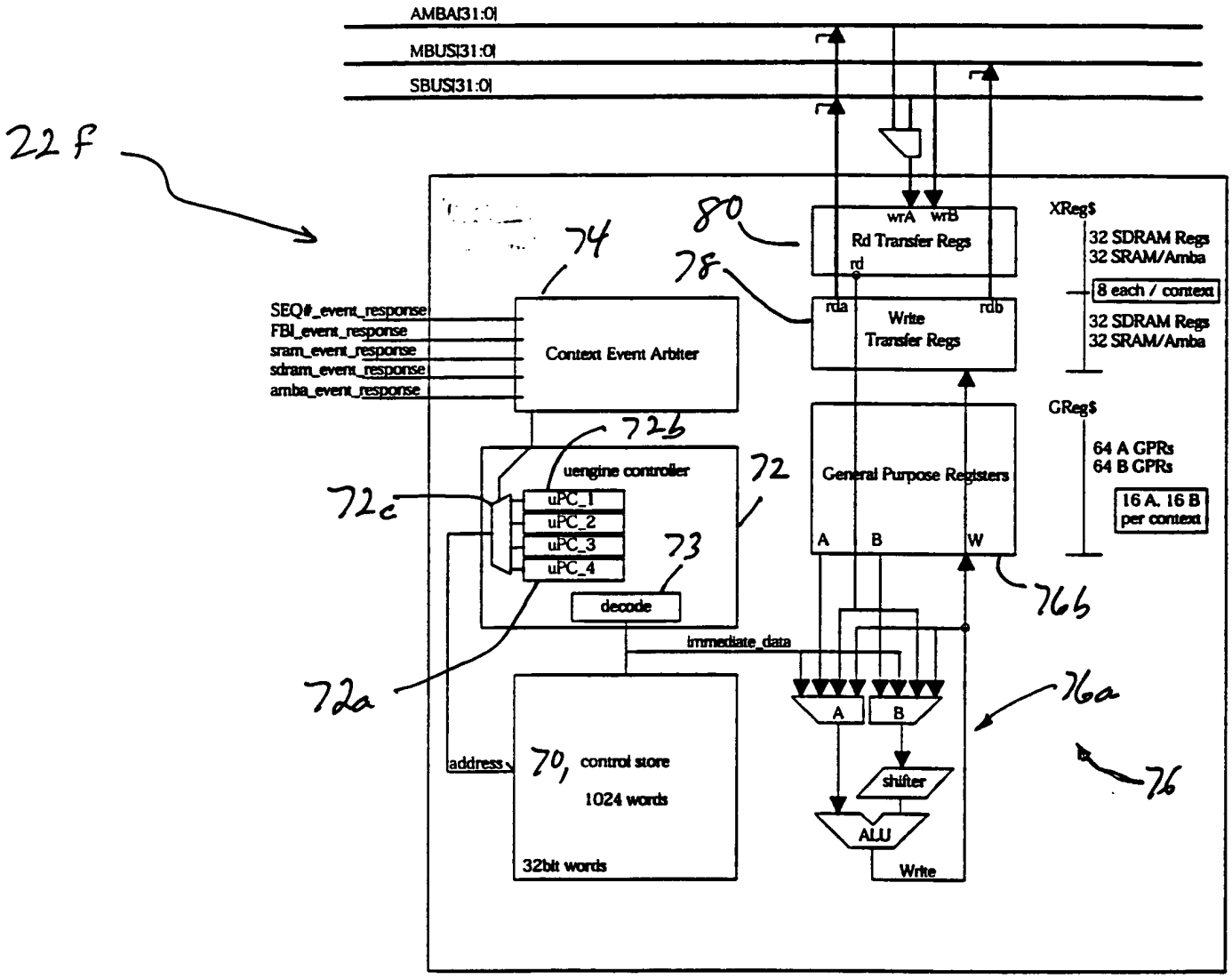

AMBA[31:0]
MBUS[31:0]
SBUS[31:0]
22f
74
80
78
wrA
wrB
Rd Transfer Regs
rd
rda
rdb
Write
Transfer Regs
XReg$
32 SDRAM Regs
32 SRAM/Amba
8 each / context
32 SDRAM Regs
32 SRAM/Amba
SEQ#_event_response
FBI_event_response
sram_event_response
sdram_event_response
amba_event_response
Context Event Arbiter
72b
uengine controller
72
72c
uPC_1
uPC_2
uPC_3
uPC_4
73
decode
General Purpose Registers
GReg$
64 A GPRs
64 B GPRs
16 A, 16 B
per context
A
B
W
76b
immediate_data
72a
76a
A
B
address
70, control store
1024 words
32bit words
shifter
ALU
Write
76

FIG. 3

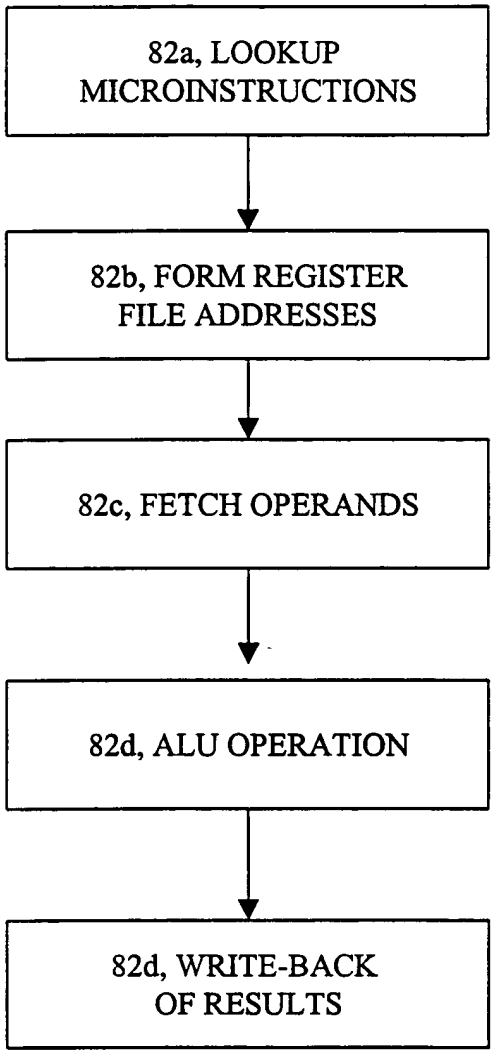
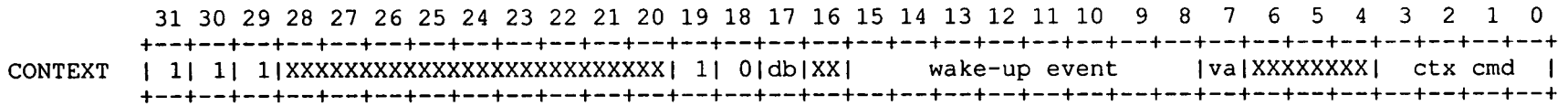

82a, LOOKUP MICROINSTRUCTIONS
82b, FORM REGISTER FILE ADDRESSES
82c, FETCH OPERANDS
82d, ALU OPERATION
82d, WRITE-BACK OF RESULTS

FIG. 3A



Context Descriptors:

1) Wake-up Events (Bits 8-15)
   - 0 = kill
   - 1 = voluntary
   - 2 = SRAM
   - 4 = SDRAM
   - 8 = FBI
   - 16 = INTER_THREAD
   - 32 = PCI_DMA_1
   - 64 = PCI_DMA_2
   - 128 = SEQ_NUM_LSB
2) db -> branch defer amount (Bit 17)
3) va -> value of sequence number (Bit 7)
4) OPCODE Bits (29-31)
5) cxt_cmd

FIG. 3*B*

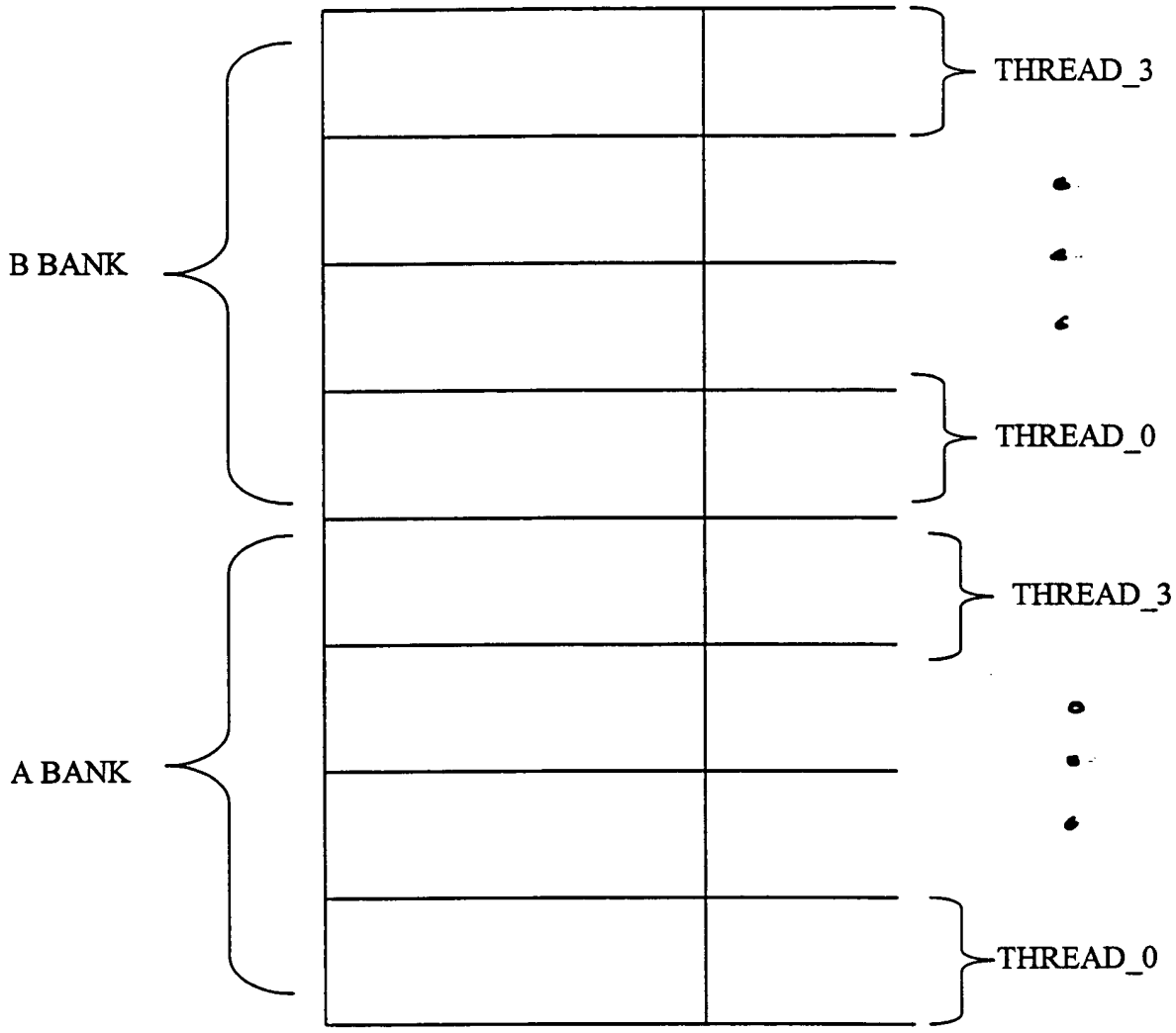


FIG. 3C

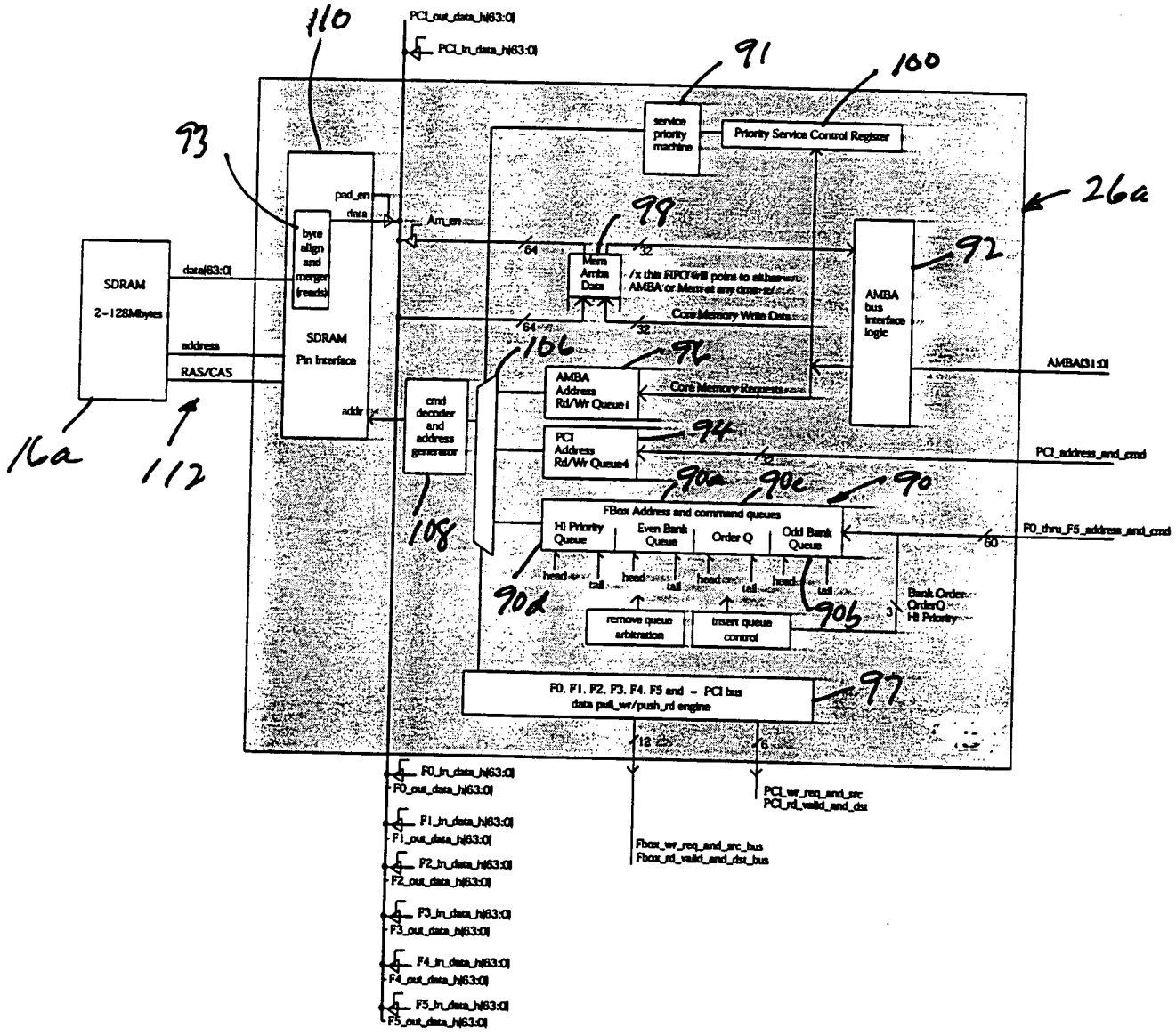

110
PCI_out_data_h[63:0]
PCI_in_data_h[63:0]
91
100
93
service priority machine
Priority Service Control Register
26a
pad_en
data
byte align and merger (reads)
Arm_en
99
64
32
SDRAM
2-128Mbytes
data[63:0]
address
RAS/CAS
SDRAM
Pin Interface
addr
Mem Amba Data
/x this FIFO will point to either AMBA or Mem at any time/
Core Memory Write Data
64
32
92
AMBA bus interface logic
AMBA[31:0]
16a
112
106
96
AMBA Address Rd/Wr Queue1
Core Memory Request
cmd decoder and address generator
94
PCI Address Rd/Wr Queue4
32
PCI_address_and_cmd
90a
90c
90
FBox Address and command queues
Hi Priority Queue
Even Bank Queue
Order Q
Odd Bank Queue
F0_thru_F5_address_and_cmd
60
108
90d
head
tail
90b
3
Bank Order
OrderQ
Hi Priority
remove queue arbitration
insert queue control
F0. F1. F2. F3. F4. F5 and - PCI bus
data pull_wr/push_rd engine
97
12
6
F0_in_data_h[63:0]
F0_out_data_h[63:0]
F1_in_data_h[63:0]
F1_out_data_h[63:0]
F2_in_data_h[63:0]
F2_out_data_h[63:0]
F3_in_data_h[63:0]
F3_out_data_h[63:0]
F4_in_data_h[63:0]
F4_out_data_h[63:0]
F5_in_data_h[63:0]
F5_out_data_h[63:0]
PCI_wr_req_and_src
PCI_rd_valid_and_dst
Fbox_wr_req_and_src_bus
Fbox_rd_valid_and_dst_bus

FIG. 4

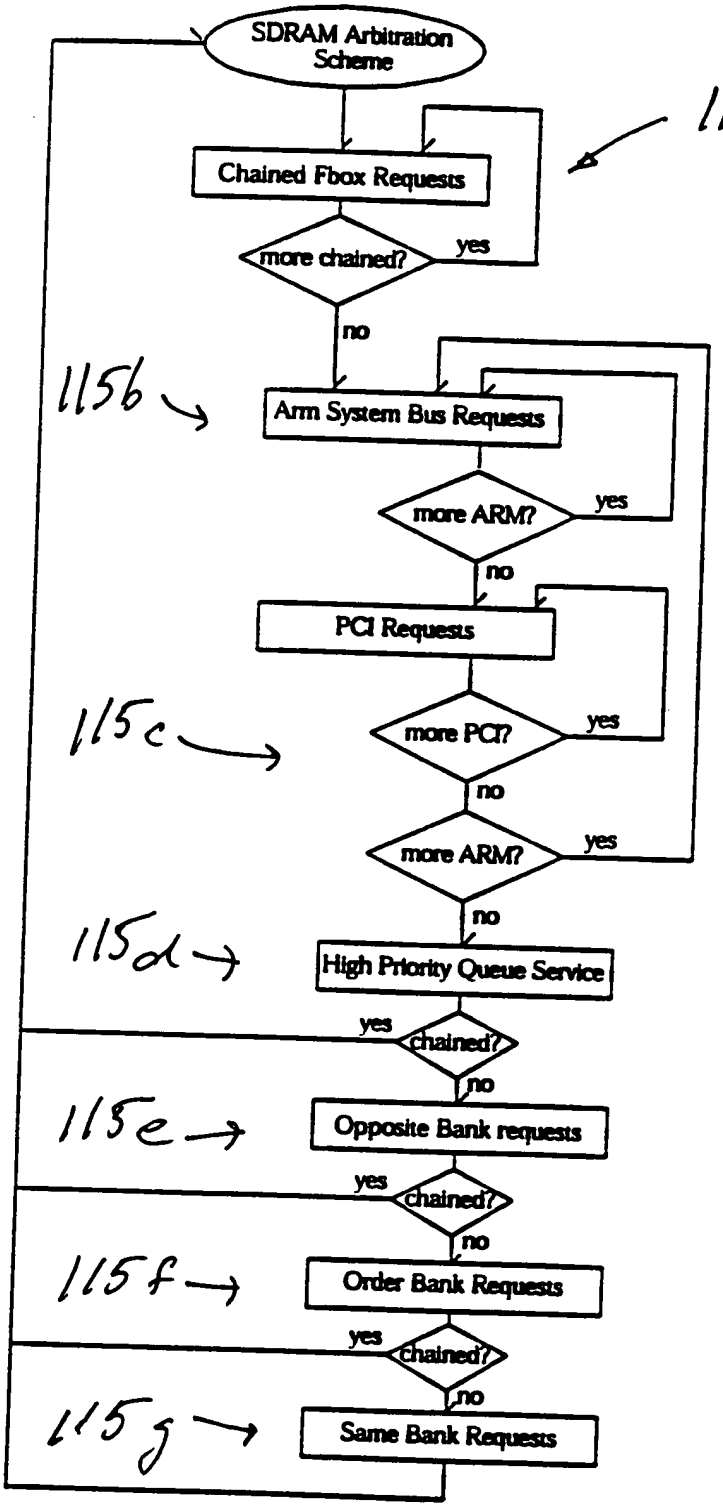
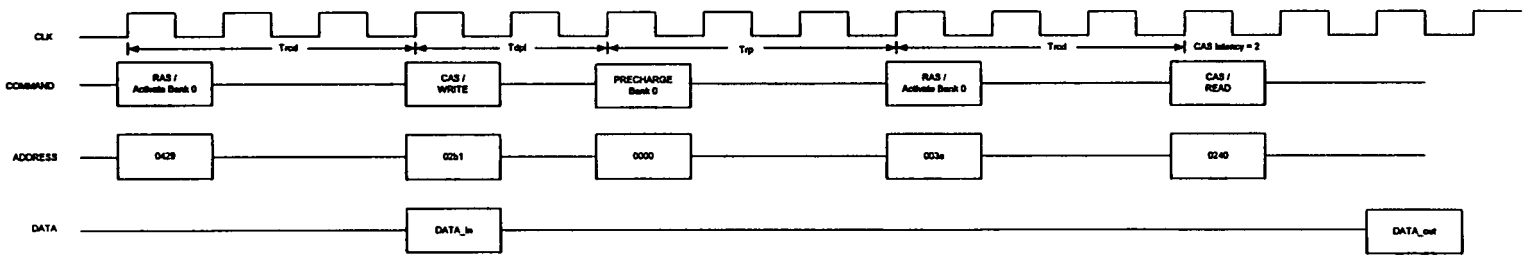

115
SDRAM Arbitration Scheme
115a
Chained Fbox Requests
more chained?
yes
no
115b
Arm System Bus Requests
more ARM?
yes
no
PCI Requests
115c
more PCI?
yes
no
more ARM?
yes
no
115d
High Priority Queue Service
yes
chained?
no
115e
Opposite Bank requests
yes
chained?
no
115f
Order Bank Requests
yes
chained?
no
115g
Same Bank Requests

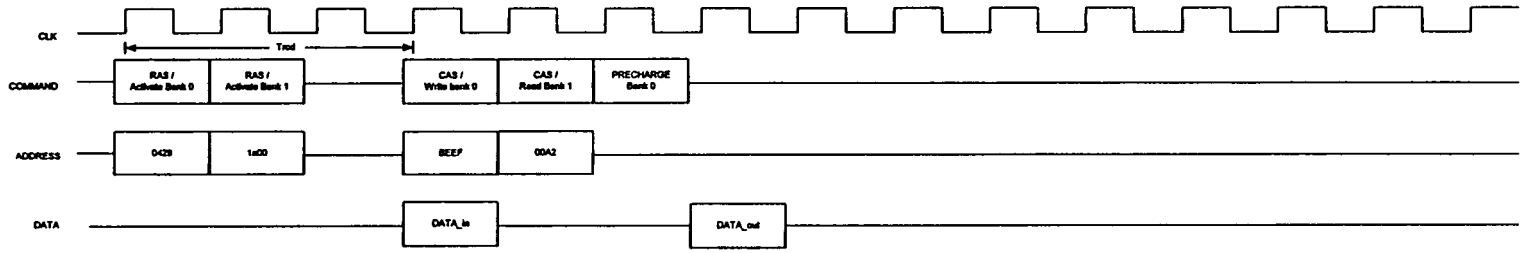
FIG. 4A

SINGLE QUADWORD WRITE FOLLOWED BY A SINGLE QUADWORD READ

without Activie Memory Optimization



with Activie Memory Optimization



where Trcd = RAS to CAS delay
Tdpl = DATA Input to Precharge Delay
Trp = Time to Precharge

FIG. 4B

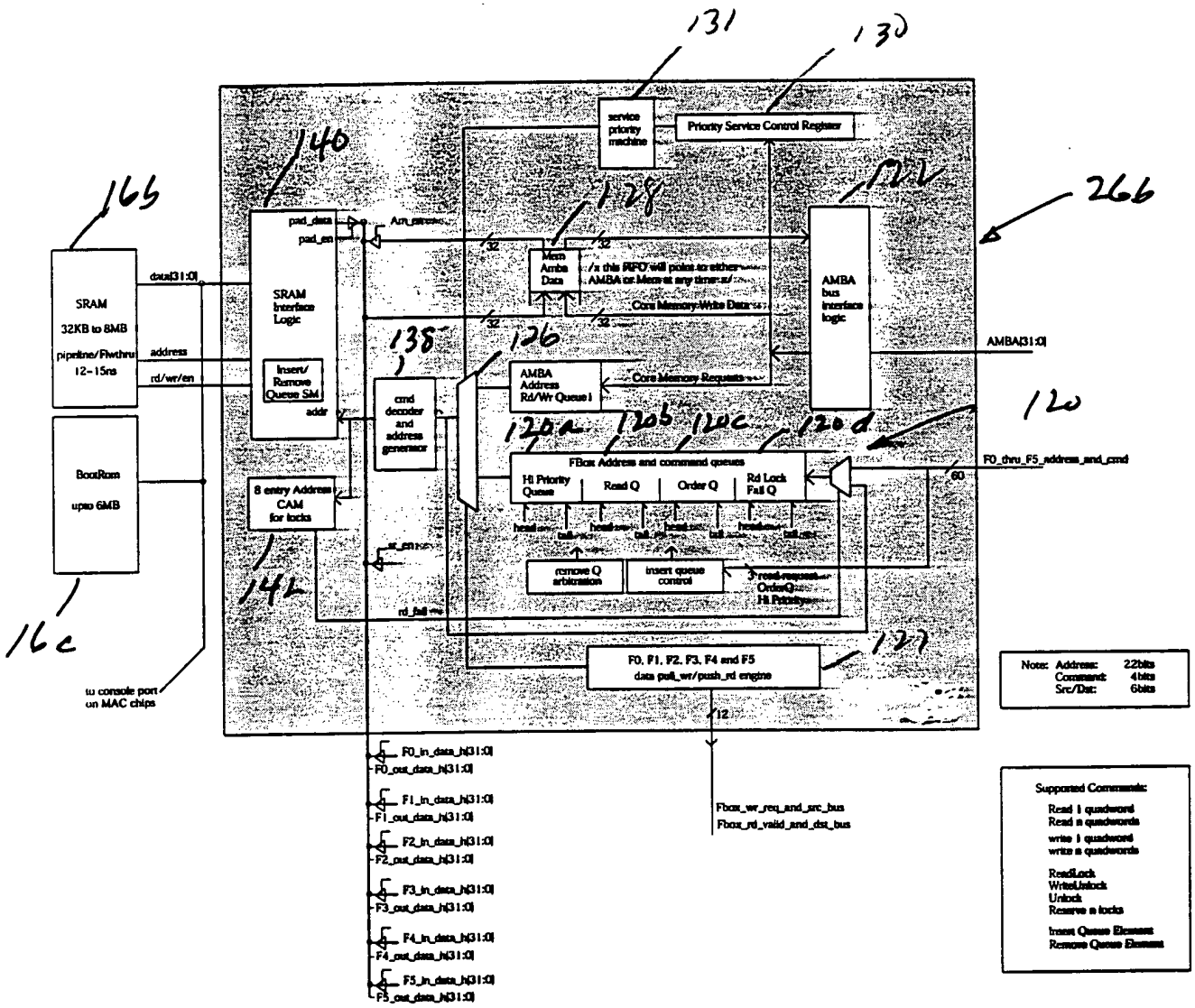
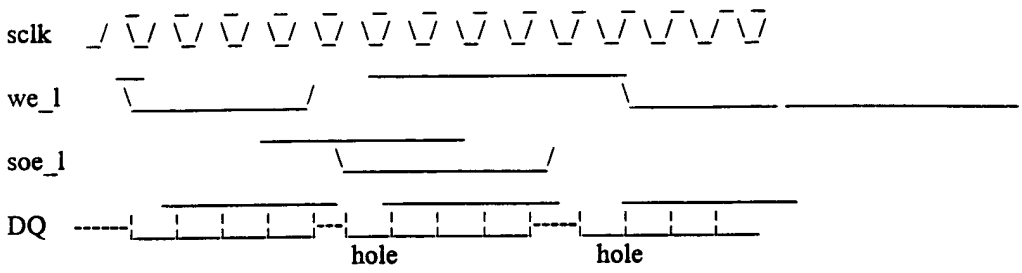
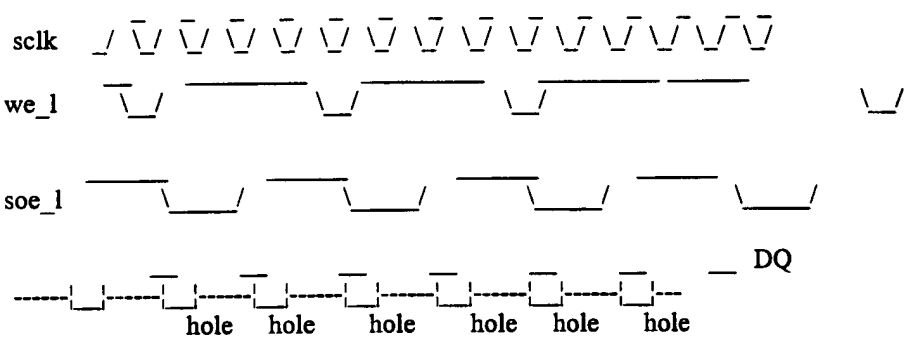

131
130
service priority machine
Priority Service Control Register
140
166
128
122
266
SRAM
32KB to 8MB
pipeline/Flwthru
12-15ns
data[31:0]
address
rd/wr/en
SRAM Interface Logic
Insert/ Remove Queue SM
addr
pad_data
pad_en
Mem Amba Data
Core Memory Write Data
AMBA bus interface logic
AMBA[31:0]
135
126
AMBA Address Rd/Wr Queue
Core Memory Requests
cmd decoder and address generator
120
120a
120b
120c
120d
FBox Address and command queues
Hi Priority Queue
Read Q
Order Q
Rd Lock Fail Q
F0_thru_F5_address_and_cmd
60
8 entry Address CAM for locks
142
remove Q arbitration
insert queue control
read-request
OrderQ
Hi Priority
BootRom
upto 6MB
16c
to console port on MAC chips
F0, F1, F2, F3, F4 and F5
data pull_wr/push_rd engine
127
Note: Address: 22bits
Command: 4bits
Src/Dst: 6bits
F0_in_data_h[31:0]
F0_out_data_h[31:0]
F1_in_data_h[31:0]
F1_out_data_h[31:0]
F2_in_data_h[31:0]
F2_out_data_h[31:0]
F3_in_data_h[31:0]
F3_out_data_h[31:0]
F4_in_data_h[31:0]
F4_out_data_h[31:0]
F5_in_data_h[31:0]
F5_out_data_h[31:0]
Fbox_wr_req_and_src_bus
Fbox_rd_valid_and_dst_bus
Supported Commands:
Read 1 quadword
Read n quadwords
write 1 quadword
write n quadwords
ReadLock
WriteUnlock
Unlock
Reserve n locks
Insert Queue Element
Remove Queue Element

FIG. 5

4 Writes and 4 Reads followed by more reads with optimization



4 Writes and 4 Reads without optimization



10 cycles vs. 14.

FIG. 5A

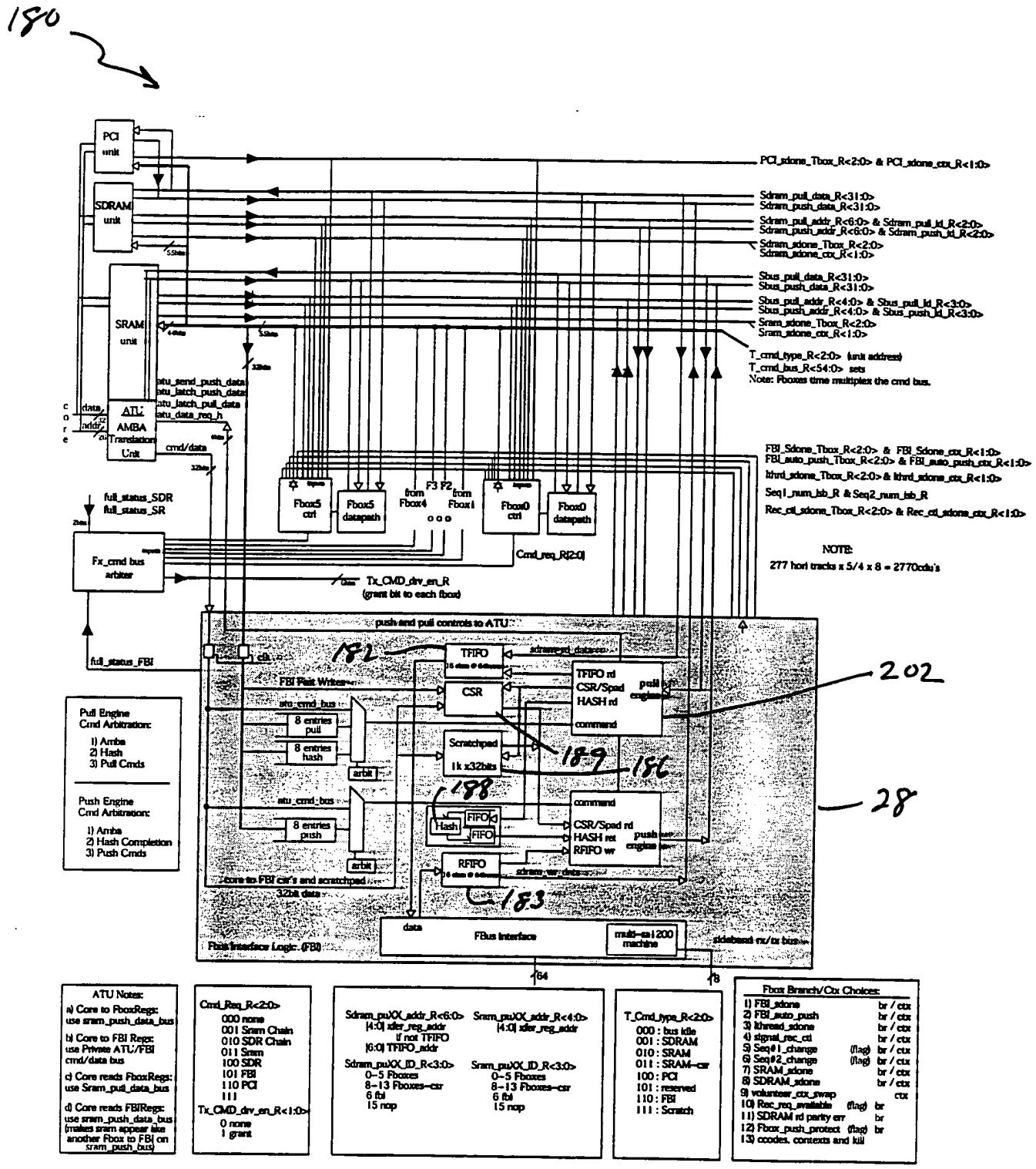

180
PCI unit
SDRAM unit
SRAM unit
ATU AMBA Translation Unit
Fx_cmd bus arbiter
Fbox5 ctrl
Fbox5 datapath
Fbox0 ctrl
Fbox0 datapath
push and pull controls to ATU
TFIFO
CSR
Scratchpad 1k x32bits
Hash
FIFO
RFIFO
TFIFO rd
CSR/Spad rd
HASH rd
command
CSR/Spad wr
HASH wr
RFIFO wr
8 entries pull
8 entries hash
8 entries push
arbit
FBI Pull Writes
core to FBI csr's and scratchpad
32bit data
FBus interface
multi-sa i200 machine
sideband rx/tx bus
182
202
189
186
188
28
183
64
18
Pull Engine Cmd Arbitration:
1) Amba
2) Hash
3) Pull Cmds
Push Engine Cmd Arbitration:
1) Amba
2) Hash Completion
3) Push Cmds
ATU Notes:
a) Core to FboxRegs: use sram_push_data_bus
b) Core to FBI Regs: use Private ATU/FBI cmd/data bus
c) Core reads FboxRegs: use Sram_pull_data_bus
d) Core reads FBIRegs: use sram_push_data_bus (makes sram appear like another Fbox to FBI on sram_push_bus)
Cmd_Req_R<2:0>
000 none
001 Sram Chain
010 SDR Chain
011 Sram
100 SDR
101 FBI
110 PCI
111
Tx_CMD_drv_en_R<1:0>
0 none
1 grant
Sdram_puXX_addr_R<6:0> |4:0| xfer_reg_addr if not TFIFO |6:0| TFIFO_addr
Sdram_puXX_ID_R<3:0> 0-5 Fboxes 8-13 Fboxes-csr 6 fbi 15 nop
Sram_puXX_addr_R<4:0> |4:0| xfer_reg_addr
Sram_puXX_ID_R<3:0> 0-5 Fboxes 8-13 Fboxes-csr 6 fbi 15 nop
T_Cmd_type_R<2:0>
000 : bus idle
001 : SDRAM
010 : SRAM
011 : SRAM-csr
100 : PCI
101 : reserved
110 : FBI
111 : Scratch
Fbox Branch/Ctx Choices:
1) FBI_sdone br / ctx
2) FBI_auto_push br / ctx
3) Ithread_sdone br / ctx
4) signal_rec_ctl br / ctx
5) Seq#1_change (flag) br / ctx
6) Seq#2_change (flag) br / ctx
7) SRAM_sdone br / ctx
8) SDRAM_sdone br / ctx
9) volunteer_ctx_swap ctx
10) Rec_req_available (flag) br
11) SDRAM rd parity err br
12) Fbox_push_protect (flag) br
13) ccodes, contexts and kill
PCI_sdone_Tbox_R<2:0> & PCI_sdone_ctx_R<1:0>
Sdram_pull_data_R<31:0>
Sdram_push_data_R<31:0>
Sdram_pull_addr_R<6:0> & Sdram_pull_id_R<2:0>
Sdram_push_addr_R<6:0> & Sdram_push_id_R<2:0>
Sdram_sdone_Tbox_R<2:0>
Sdram_sdone_ctx_R<1:0>
Sbus_pull_data_R<31:0>
Sbus_push_data_R<31:0>
Sbus_pull_addr_R<4:0> & Sbus_pull_id_R<3:0>
Sbus_push_addr_R<4:0> & Sbus_push_id_R<3:0>
Sram_sdone_Tbox_R<2:0>
Sram_sdone_ctx_R<1:0>
T_cmd_type_R<2:0> (unit address)
T_cmd_bus_R<54:0> sets
Note: Fboxes time multiplex the cmd bus.
FBI_Sdone_Tbox_R<2:0> & FBI_Sdone_ctx_R<1:0>
FBI_auto_push_Tbox_R<2:0> & FBI_auto_push_ctx_R<1:0>
Ithrd_sdone_Tbox_R<2:0> & Ithrd_sdone_ctx_R<1:0>
Seq1_num_lsb_R & Seq2_num_lsb_R
Rec_ctl_sdone_Tbox_R<2:0> & Rec_ctl_sdone_ctx_R<1:0>
NOTE:
277 hori tracks x 5/4 x 8 = 2770cdu's
full_status_SDR
full_status_SR
full_status_FBI
Cmd_req_R[2:0]
Tx_CMD_drv_en_R (grant bit to each fbox)
F3 F2 from Fbox4 from Fbox1 o o o

FIG. 6